



Scalable Heterogeneous Computing: Accelerating MPI & I/O on CPU+GPU Nodes

Sadaf Alam

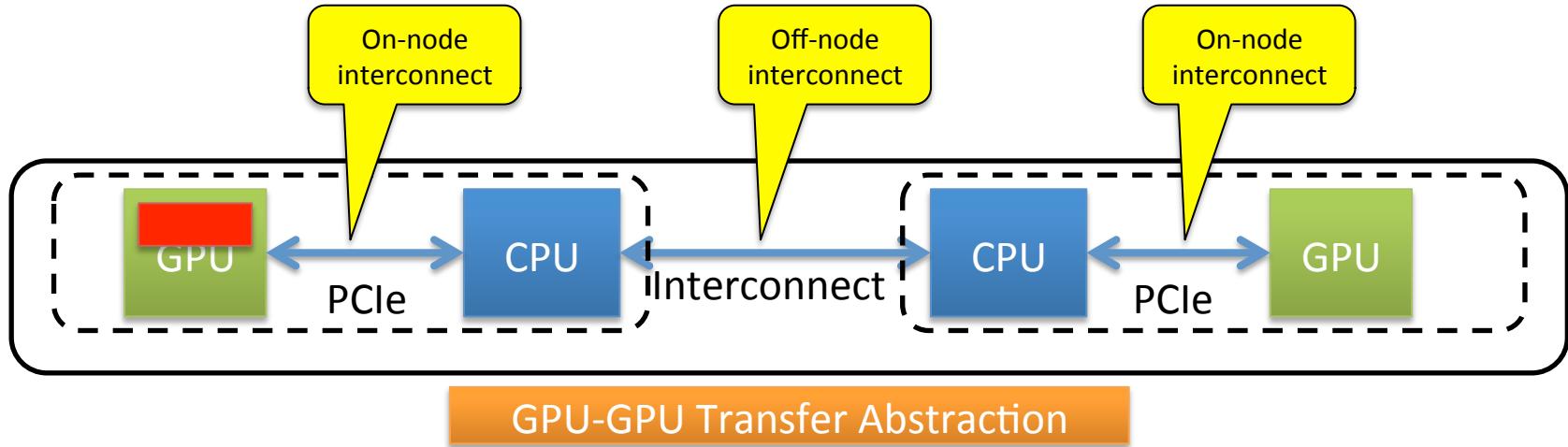
Swiss National Supercomputing Centre

SOS16 Workshop

Collaborators & contributors

- DK Panda, Devendar Bureddy
Ohio State University
- Antonio J. Peña
Technical University of Valencia
- Oliver Fuhrer
MeteoSwiss

Characteristics of Internode GPU Communication



Copy data from GPU to CPU (device to host)

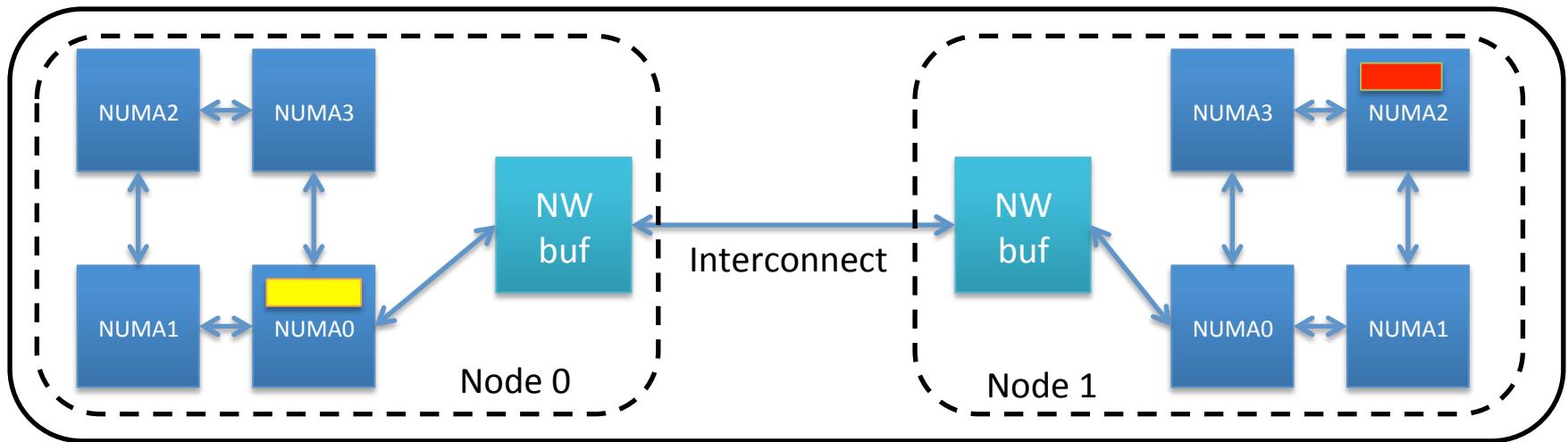
MPI message transfer between two CPUs (host to host MPI)

Copy data from CPU to GPU (host to device)

Motivating Examples

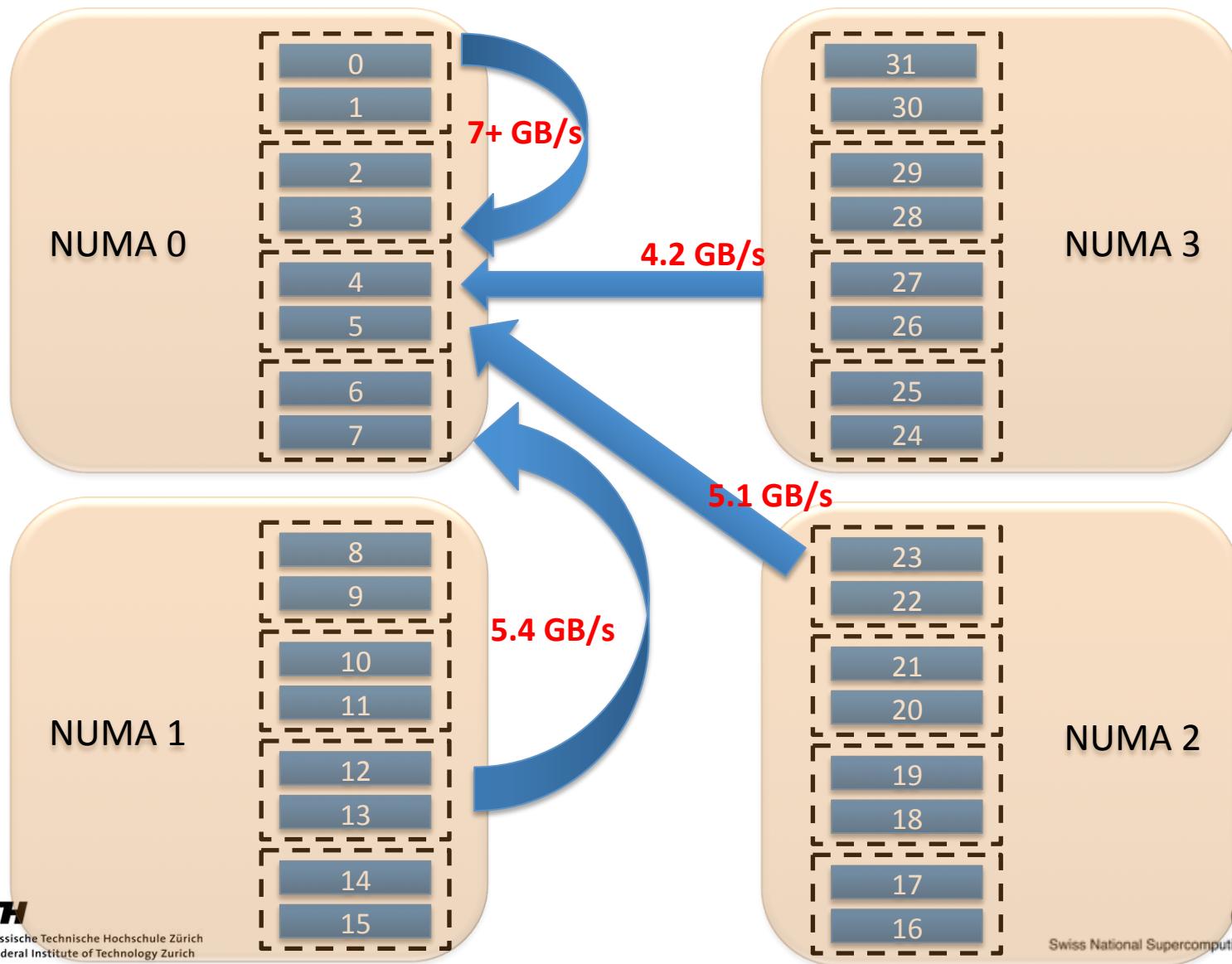
- GPU-accelerated weather modeling application (COSMO)
 - Halo Exchanges between GPUs (on-node and off-node)
 - Different halo sizes
 - Goal: a single source, auto-tuned, code base for platforms with diverse CPU/GPU node configurations
- GPU virtualization (rCUDA)
 - Solution for small to mid-size heterogeneous clusters
 - Scheduling on remote GPU without performance penalties
 - Goal: define a cost model for scheduling decisions through detailed characterization of GPU-GPU data transfers

Bandwidth and Latency Considerations on Homogenous Multi-core Systems

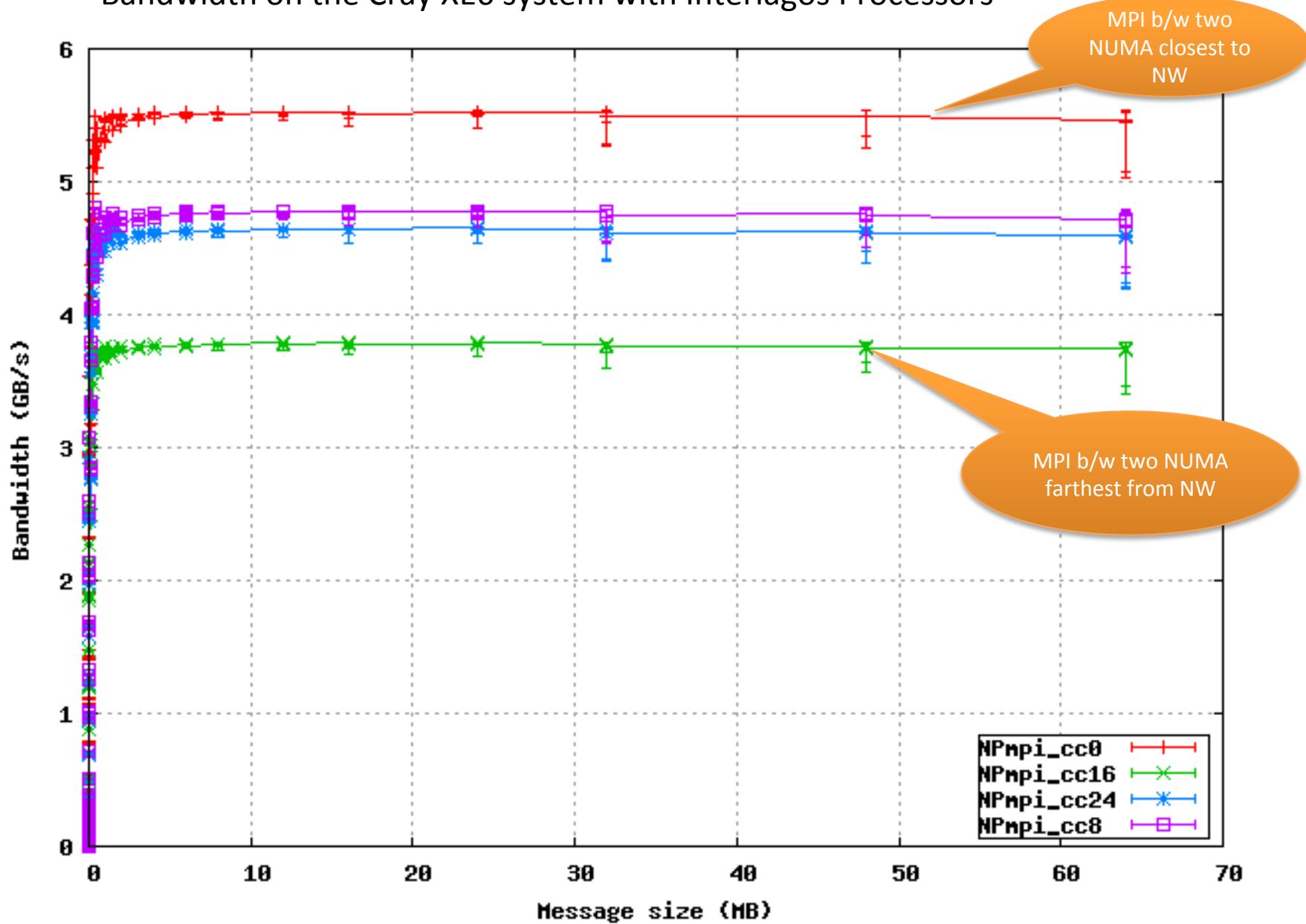


CPU-CPU Transfer Abstraction with Non-Uniform
Memory Access (NUMA) considerations

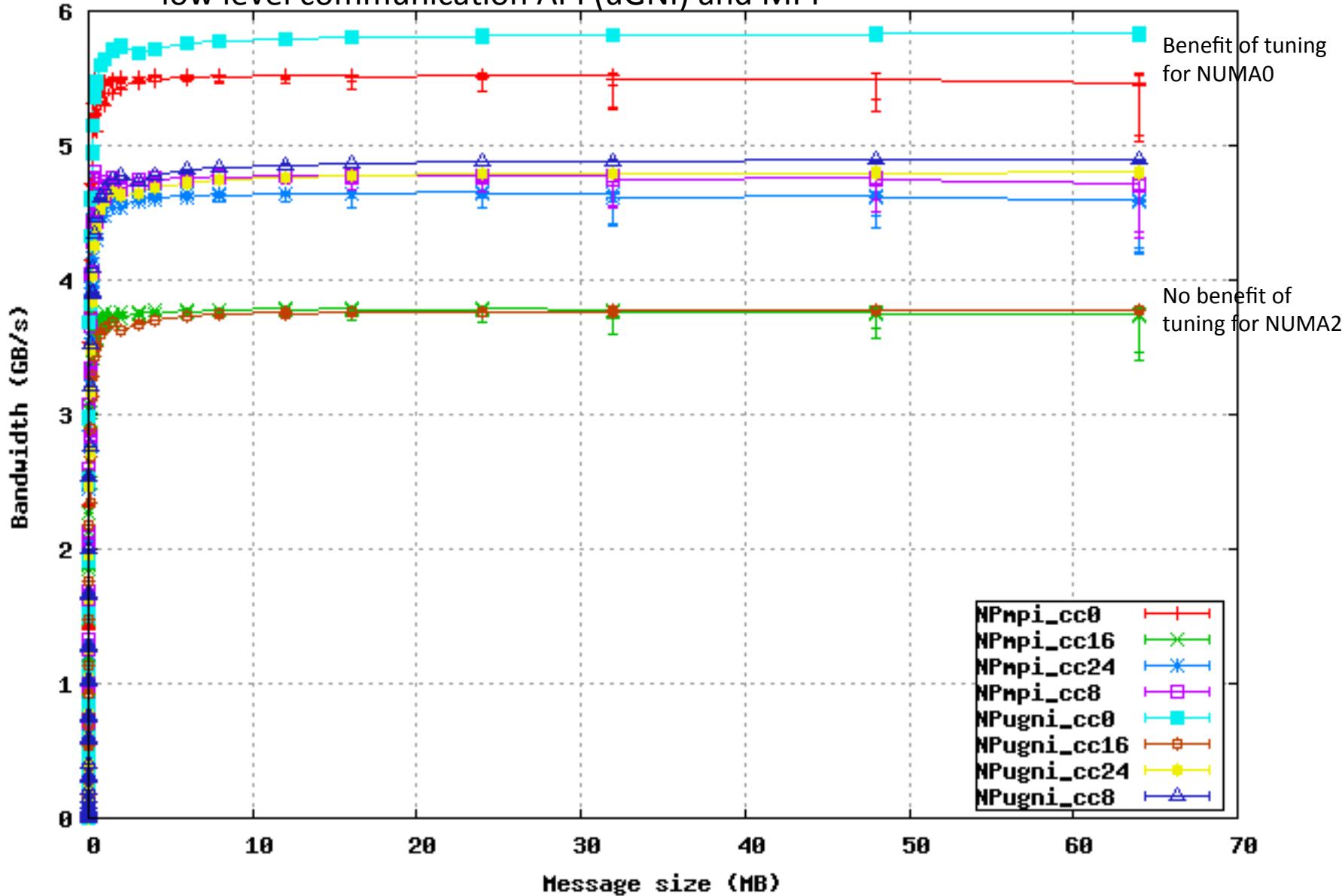
Empirical Results (One, Dual-socket Interlagos)



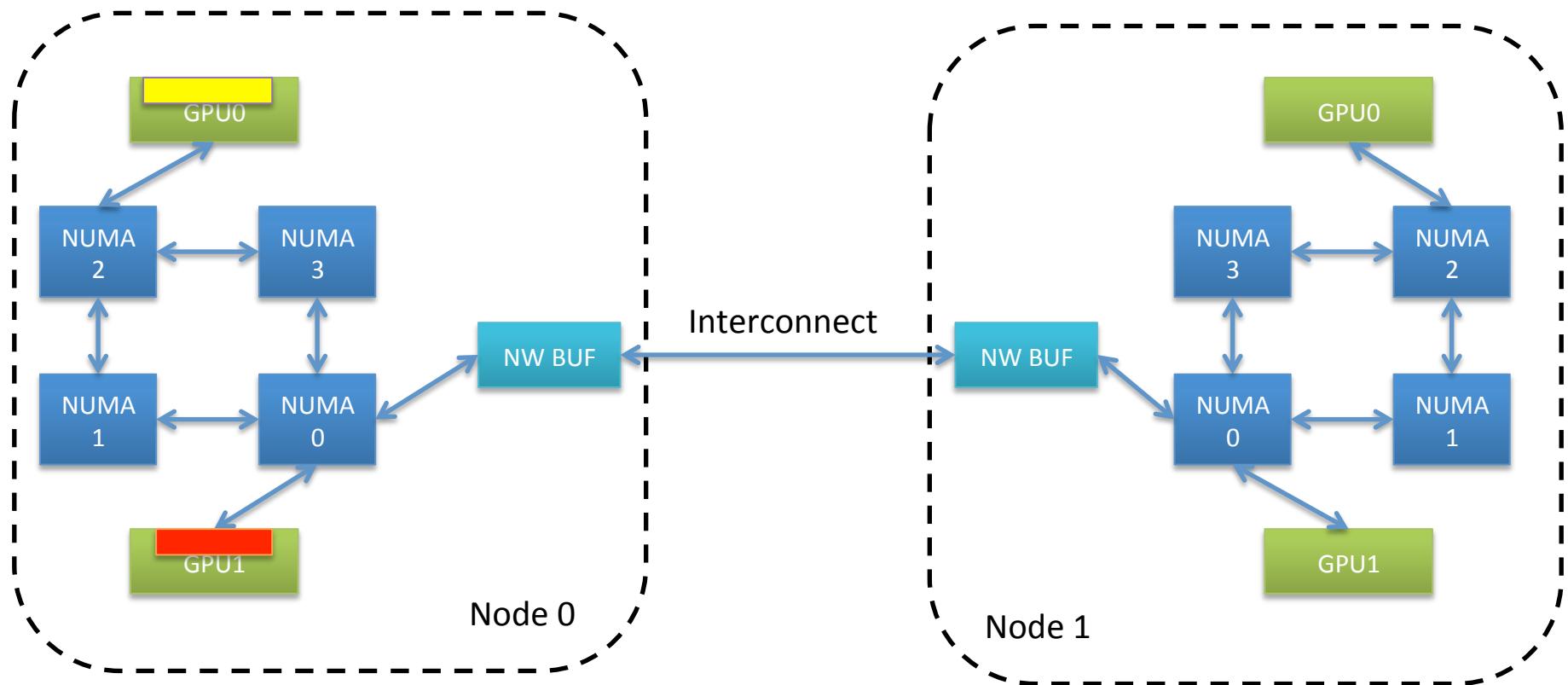
Bandwidth on the Cray XE6 system with Interlagos Processors



Bandwidth on the Cray XE6 system with Interlagos Processors using low level communication API (uGNI) and MPI



Latency and Bandwidth: CPU & Accelerators



Implementation Scenarios

Naïve implementation

$$Latency_{total} = L_{\substack{CPU-GPU \\ \text{pageable}}} + L_{\substack{GPU-CPU \\ \text{pageable}}} + L_{\substack{CPU-CPU \\ MPI}}$$

Typical implementation

$$Latency_{total} = L_{\substack{CPU-GPU \\ pinned}} + L_{\substack{GPU-CPU \\ pinned}} + L_{\substack{CPU-CPU \\ MPI}}$$

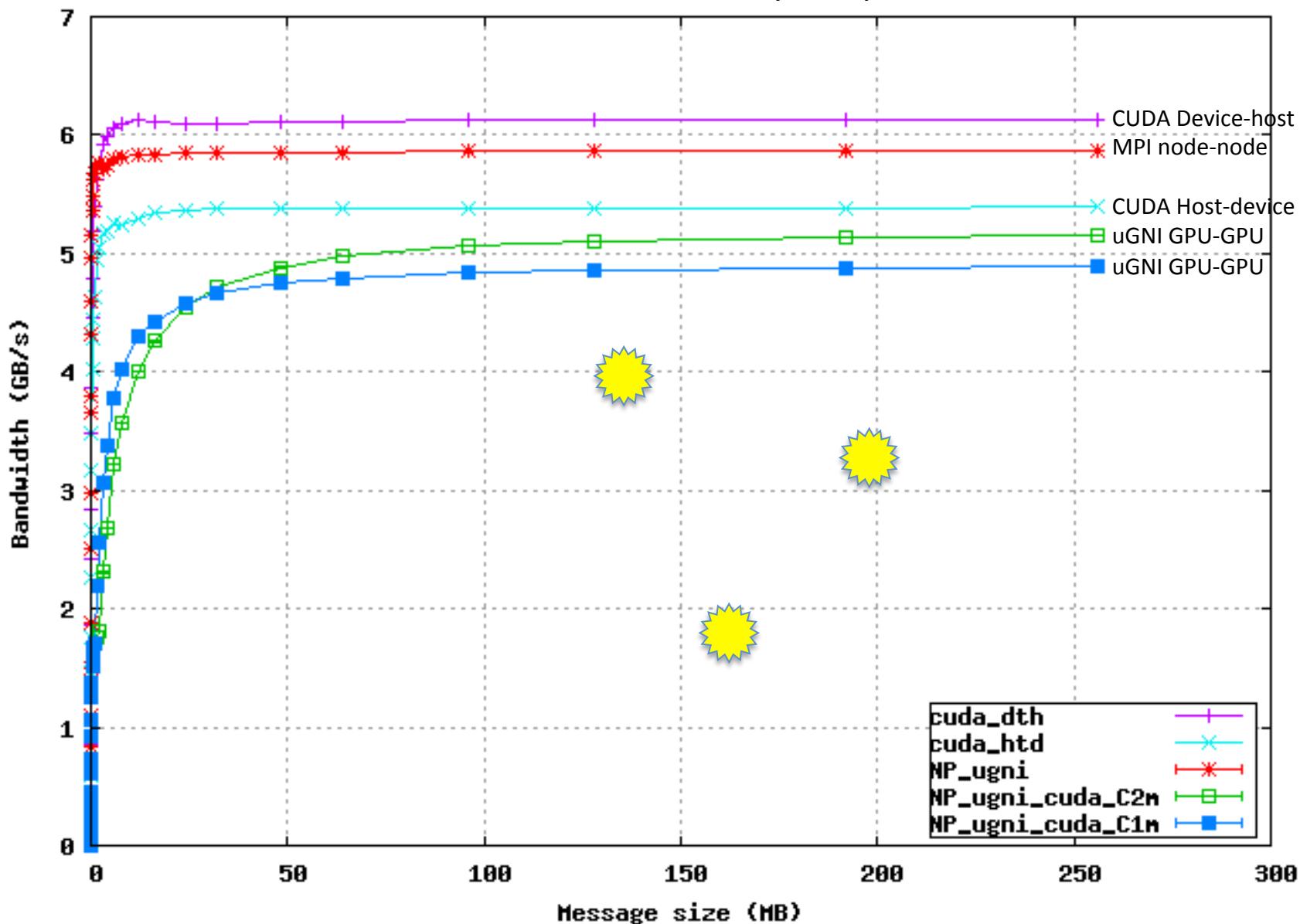
Advanced implementation

$$Latency_{total} = MIN\left(L_{\substack{CPU-GPU \\ pinned}}\right) + MIN\left(L_{\substack{GPU-CPU \\ pinned}}\right) + MIN\left(L_{\substack{CPU-CPU}}\right)$$

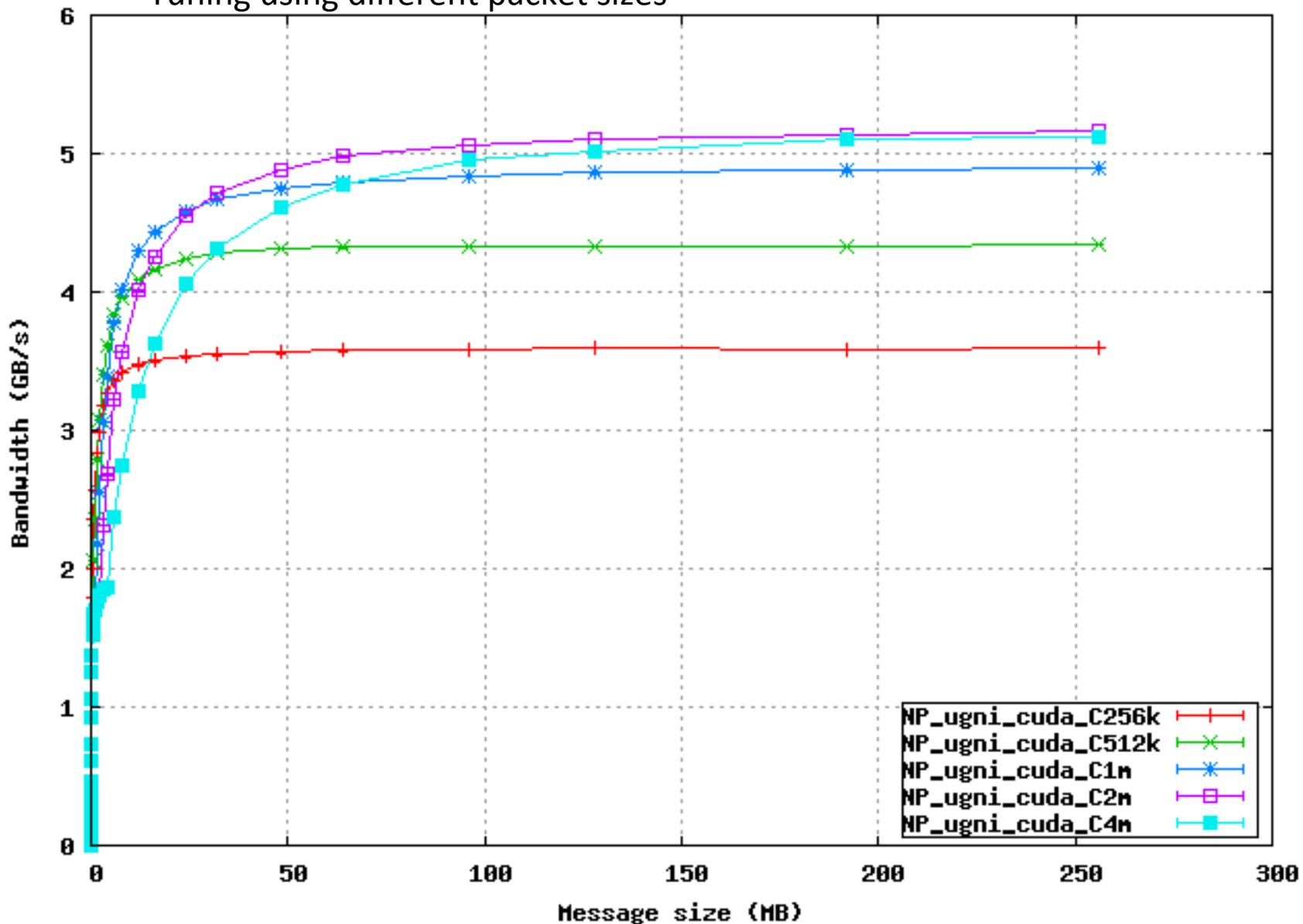
Expert implementation

$$Latency_{total} = MAX\left\{MIN\left(L_{\substack{CPU-GPU}}\right), MIN\left(L_{\substack{GPU-CPU}}\right), MIN\left(L_{\substack{CPU-CPU}}\right)\right\}$$

Bandwidth between two GPU nodes on Cray XK6 platform



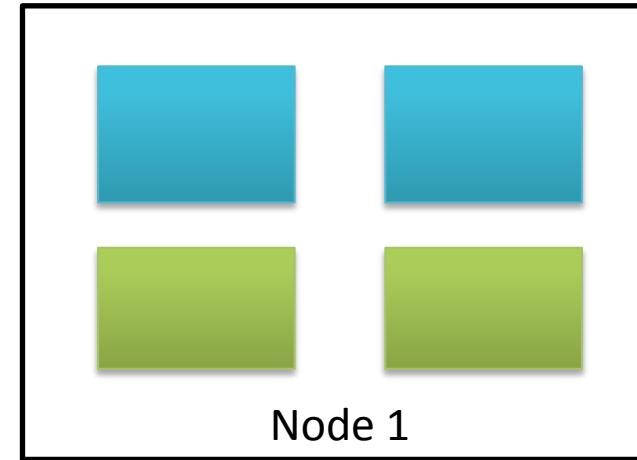
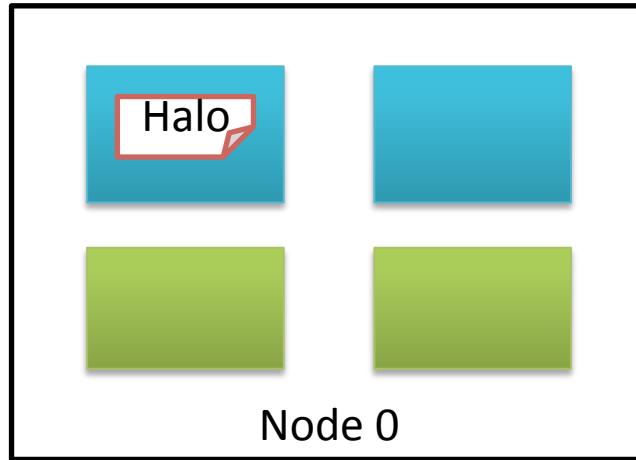
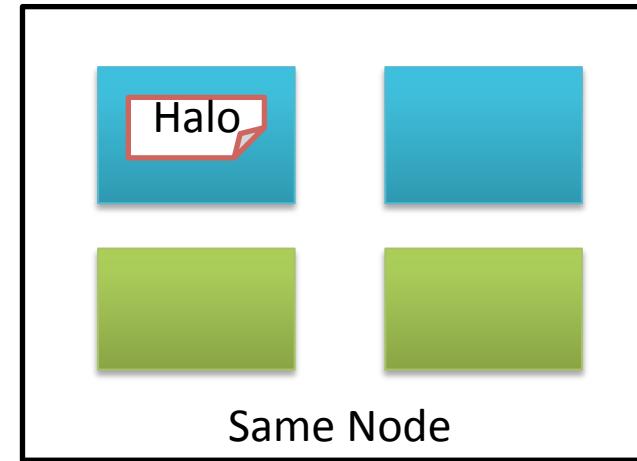
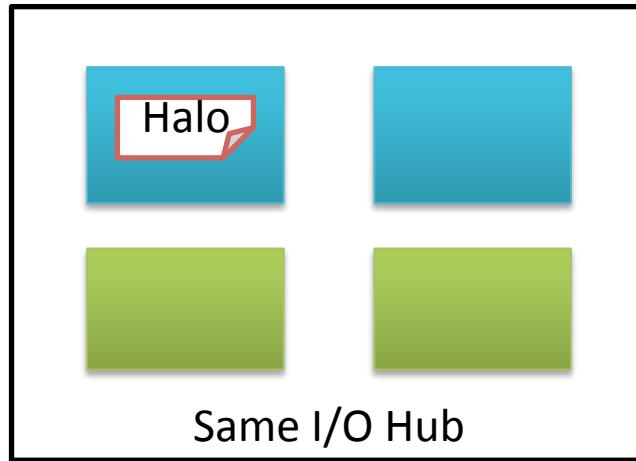
Bandwidth between two GPU nodes on Cray XK6 (Tödi) platform
Tuning using different packet sizes



Advances in CUDA and InfiniBand Drivers

- Enabling of shared access to device memories
- Uniform virtual addressing
- Peer to peer support
- GPU and CPU bindings
- An optimal implementation exploits all of the above
BUT there shouldn't be a rewrite after the introduction of a new feature

Halo Exchange Code Implementation with MVAPICH2-GPU



Single code base for all configurations

Implementation using MVAPICH2-GPU

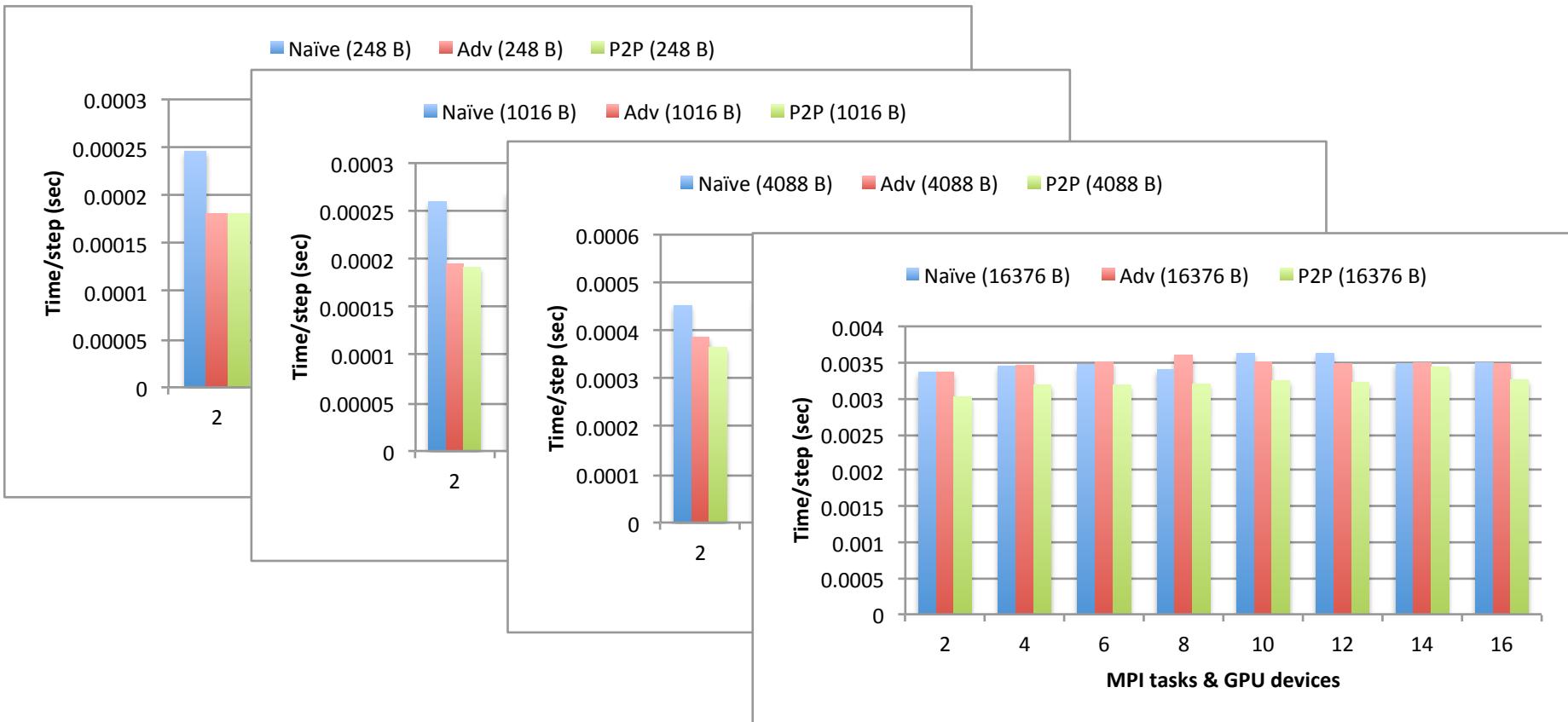
| | Naïve | Advanced | GPU only |
|--|-------|----------|----------|
| Buffer initialization and final copy on the host | ✓ | ✓ | ✗ |
| Copy buffers to and from host for each time step | ✓ | ✗ | ✗ |
| Computation on the device | ✓ | ✓ | ✓ |
| Non-blocking sends and receives | ✓ | ✓ | ✓ |

Lower memory, & fewer lines of code

Least host memory & fewest lines of code

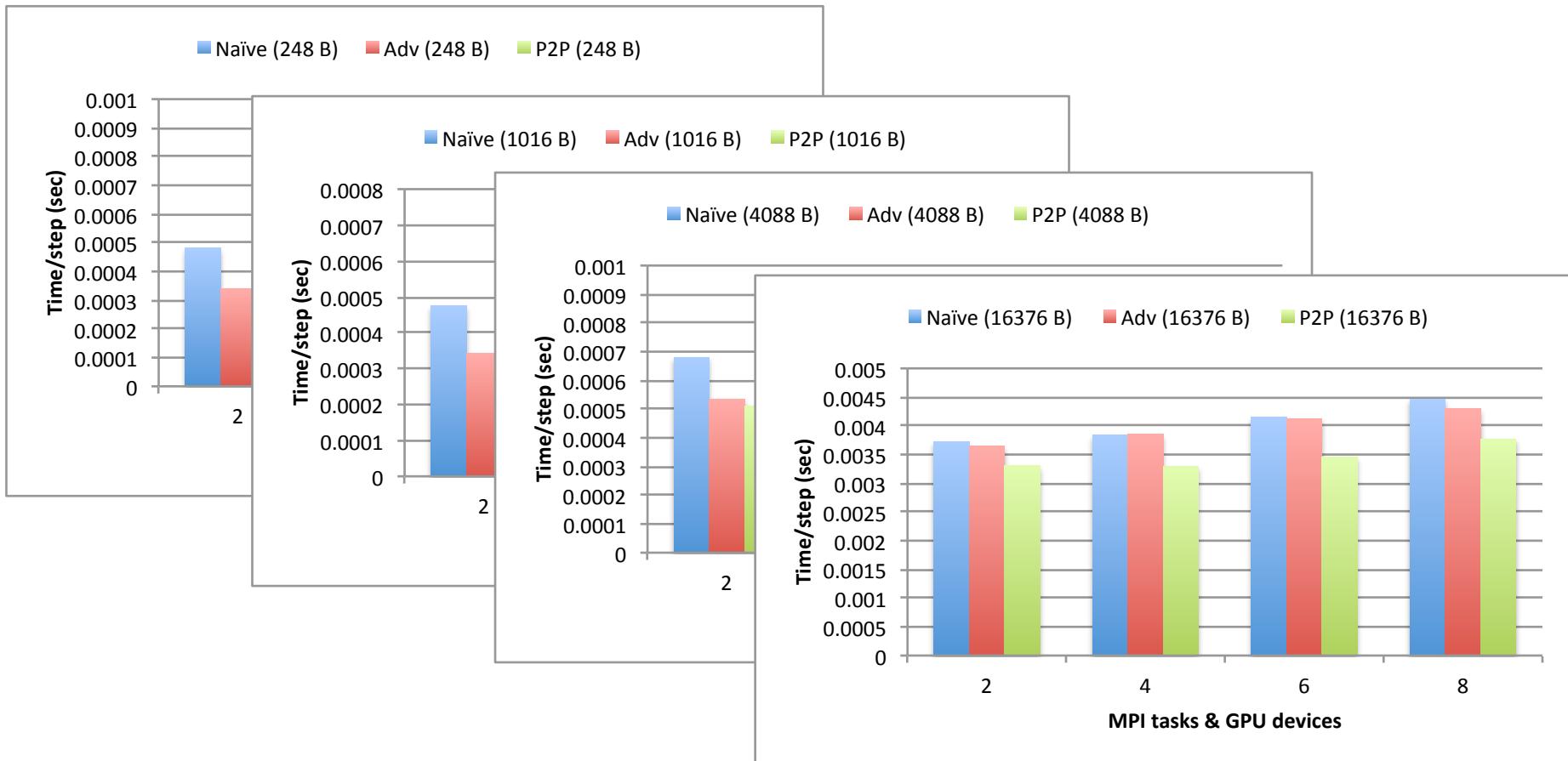
Results (MPI + GPU Halo Exchange)

- On a dual-socket, dual-GPU, ID QDR system



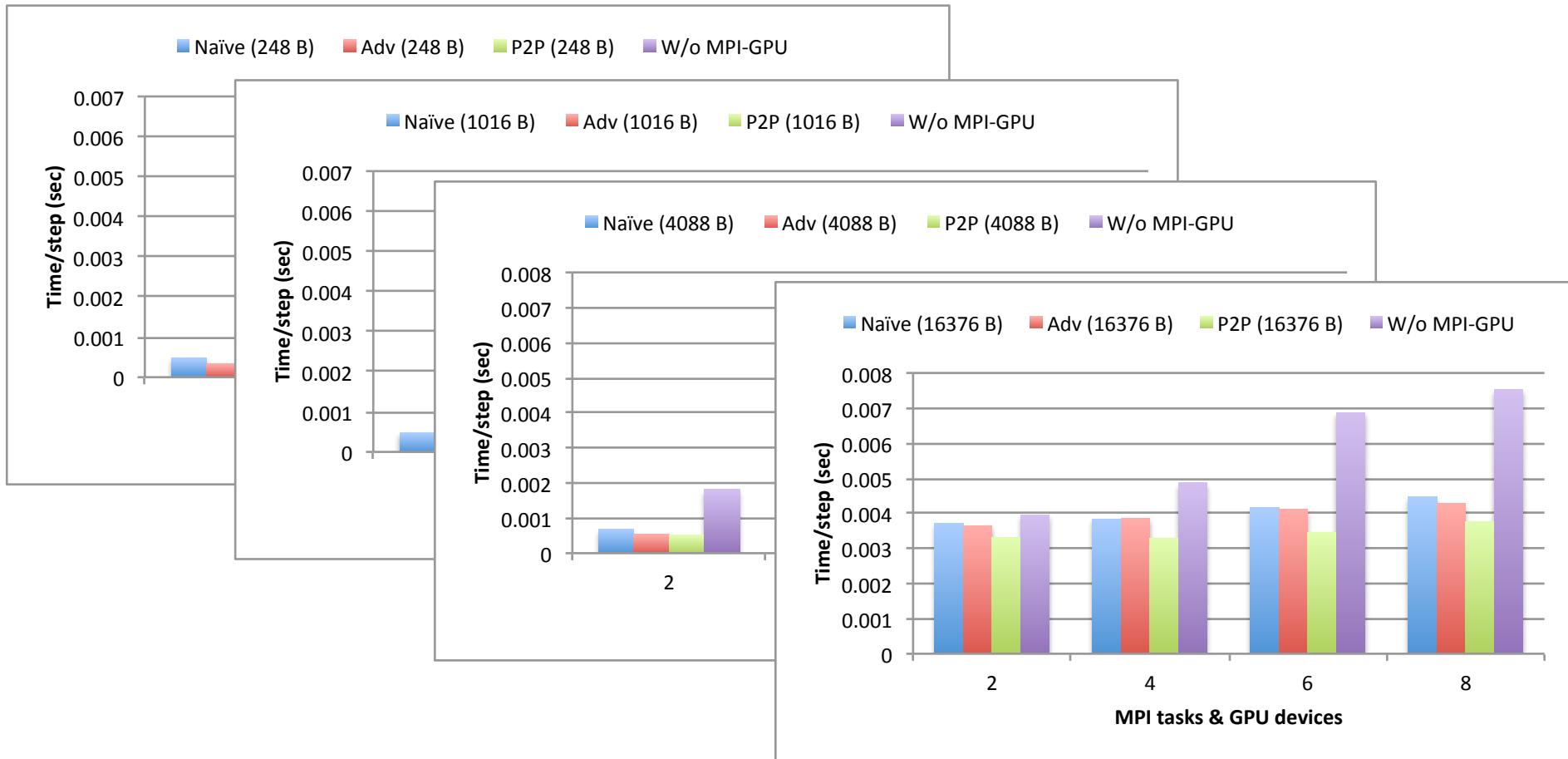
Results (MPI + GPU Halo Exchange)

- On a 8 GPU, single node system with 2 I/O hubs



Results (MPI + GPU Halo Exchange)

- Without MVAPICH2-GPU



Shortcoming of Current Solution

- Not a portable solution
 - Only a single MPI library implementation
 - Dependency on a given runtime & driver
 - Network hardware dependency
- Interoperability constraints
 - Emerging, directive-based programming
 - MPI standards conformance

Future Plans & Thoughts

- Move additional computation to the GPU to minimize transfers
 - Towards a self hosted implementation
 - Reduction in host memory requirements
- Encourage middleware developers to provide similar solutions
 - OpenCL support
 - Multiple accelerators support
 - Multiple MPI implementation support (discussions in the forum)
- Explore interoperability with emerging programming models for accelerators